HDS Serenity Ledger

Pedro Barão, *98963, METI* Martim Abreu,*98956*, *METI*
Afonso Gomes, *98921, LETI*

Abstract

This project aims to develop HDS Serenity (HDS2), a permissioned blockchain system with high dependability guarantees. In the second stage, goals include providing a cryptocurrency application, enhancing Byzantine fault tolerance, and evaluating correctness and performance. The cryptocurrency application allows clients to transfer funds securely, ensuring non-negative balances, unauthorised modification prevention, and operation non-repudiation. Byzantine fault scenarios are analysed, and appropriate countermeasures are discussed and implemented. Implementation steps involve ensuring compliance with initial requirements, extending support for Byzantine clients, defining application-level operations, and implementing robust testing against Byzantine behaviour.

1. Overview

The HDS Serenity (HDS2) project aims to develop a highly dependable permissioned blockchain system, with the goal of refining its application semantics and dependability guarantees in its second stage. This project focuses on several key objectives.

1.1. Cryptocurrency Application

The objective is to implement a cryptocurrency application within the blockchain system. This application allows clients to transfer funds securely between accounts, ensuring non-negative balances, prevention of unauthorised modifications, and non-repudiation of operations.

1.2. Enhanced Byzantine Fault Tolerance

Another critical goal is to strengthen the system's Byzantine fault tolerance, including the ability to tolerate Byzantine clients. This involves analysing potential attacks from Byzantine servers or clients, implementing appropriate countermeasures, and demonstrating the system's resilience to various Byzantine attacks.

1.3. Experimental Confirmation

The project aims to experimentally confirm the correctness, performance, and resilience under attacks of the implementation. It is essential to ensure that a single malicious replica cannot significantly slow down the entire system.

Overall, the HDS2 project represents a comprehensive effort to refine a permissioned blockchain system, addressing critical challenges such as ensuring security, dependability, and resilience against Byzantine faults. Through careful design, implementation, and testing, the project aims to advance the state of the art in permissioned blockchain technology.

2. Most Relevant Implementations Steps

2.1. Round Change

In the IBFT protocol used by the Quorum blockchain, a round change occurs when the network can't reach consensus due to issues like leader failure or network delays. This mechanism kicks in to keep the blockchain moving smoothly, ensuring that transactions, including transfer messages, don't get stuck. When a node's timer expires without achieving consensus, it triggers a round change, suggesting the network might need a new leader or different conditions to agree on the next block's content.

Nodes then communicate, sharing their highest agreed-upon transaction information and proposing what should be included in the blockchain next. This could be a transfer message that was ready but not yet committed due to the hiccup. A new leader is chosen based on the round number, and this leader proposes the next block's content, potentially re-proposing the transfer message if it was the subject of consensus in the failed round.

The network aims to reach a new consensus on this proposal, following its normal process of preparing, committing, and finally adding the proposed transactions to the blockchain. This round change process ensures that even in the face of disruptions, the network can recover and continue adding new transactions to the blockchain, maintaining its integrity and operation.

2.2. Check Balance Operation

The implementation of the check balance operation lies at the core of the cryptocurrency application within the blockchain system. This operation serves as a fundamental tool for users to ascertain the current balance associated with their respective wallet addresses. By facilitating this process, the system fosters transparency and trust among participants.

The operation entails a systematic process on both the server and client sides. When a user requests a balance check, the client dispatches requests to all servers in the network. Each server, upon receiving the request, decodes the provided public key and calculates the balance associated with it. Depending on predefined conditions such as leadership status or specific tests, servers generate responses containing the calculated balances. These responses are then transmitted back to the client.

On the client side, responses from servers are collected and analysed. The system waits until a sufficient number of responses, known as a quorum, are received. Once the quorum is reached, the client logs the balance of the requested wallet. To ensure accuracy, the client considers the most common response among received messages.

Throughout the implementation process, rigorous testing is conducted to validate the correctness and robustness of the operation. This includes testing under various conditions such as network failures and unexpected server behaviours. By ensuring the operation's fault tolerance and reliability, the system enhances transparency and trust among users in the blockchain network.

In essence, the check balance operation is pivotal for maintaining the integrity and functionality of the cryptocurrency application. Its seamless execution, coupled with thorough testing, reinforces the system's reliability and fosters a sense of confidence among users.

2.3. Transfer Operation

The transfer operation is a fundamental component of any cryptocurrency application within a blockchain system, facilitating the transfer of digital assets between users' wallet addresses.

Similar to the check balance operation, the transfer operation involves a coordinated process between the client and servers within the network. When a user initiates a transfer request, the client constructs a transaction containing essential details such as sender and recipient addresses and the amount to be transferred.

When a client wishes to transfer a certain quantity of coins, they initiate the transfer operation by specifying the destination and the amount to be transferred. Upon performing a transaction request, the client broadcasts it to all servers in the network (Transfer Message). Upon receipt of a TransferMessage from a peer node, the algorithm meticulously scrutinises the validity of the transaction.

The recipient node examines the transfer amount, ensuring it does not exceed the sender's available balance. Furthermore, the node verifies whether the sender possesses the authority to initiate the transfer.

Should the transfer amount exceed the sender's balance or if the sender lacks authorization, the recipient promptly dispatches an appropriate response (TransferResponseMessage) to the sender, signalling the transaction's failure.

Upon satisfying the preliminary validation checks, the recipient node embarks on initiating a consensus process for the transfer. The node invokes the startConsensus method, signalling the commencement of a consensus instance tailored to adjudicate the validity and sequencing of the transfer. The consensus process unfolds through a series of orchestrated steps, fostering agreement among network peers regarding the legitimacy of the transfer. Nodes engage in a synchronised exchange of messages, spanning diverse phases such as PRE-PREPARE, PREPARE, and COMMIT. This orchestrated dialogue fosters a convergence of opinions regarding the transfer's veracity. Through iterative communication and deliberation, nodes endeavour to achieve consensus, culminating in a unanimous accord on the validity of the transfer.

With consensus duly established, the recipient node undertakes the irreversible commitment of the transfer, thereby effectuating its inclusion in the ledger. The node meticulously updates its ledger, capturing the pertinent details of the transfer. Simultaneously, it orchestrates the requisite adjustments to the sender's and recipient's balances, ensuring coherence and integrity across the ledger.

After the ledger update, the node disseminates a response (TransferResponseMessage) to the original sender, furnishing insights into the outcome of the transfer operation.

The designated server leader will receive a fixed tax (one coin) for facilitating each transfer.

In the context of the transfer operation, upon initiating a transfer request, the client awaits responses from participating nodes in the network. The system remains in a state of expectancy until a sufficient number of responses, constituting a quorum, are amassed. Once this quorum is achieved, the client meticulously examines the received responses to ascertain the outcome of the transfer. To

ensure accuracy and consistency, the client relies on the most prevalent response among the received messages, thereby adjudging the success or failure of the transfer operation.

This streamlined process not only ensures the integrity and security of the transfer but also provides a seamless experience for the client, who receives timely notification of the transaction's completion. By employing a consensus algorithm like IBFT, the system can reliably handle transfer operations while maintaining robustness and efficiency in its operations.

2.4. Client Byzantine Fault Tolerance

Enhancing client Byzantine fault tolerance is essential for fortifying the resilience of the blockchain system against malicious client behaviour. This step involves implementing strict measures within the system architecture to prevent unauthorised interactions and ensure protocol adherence. By restricting client messaging to specific actions, such as appending data, and implementing mechanisms to mitigate Byzantine behaviour, the system enhances its overall security posture.

2.5. Byzantine Behaviour Testing

Thoroughly testing the system's behaviour under Byzantine faults is critical for validating its dependability guarantees. This step focuses on designing and executing a comprehensive set of experiments to assess the system's resilience to various Byzantine attacks, including attacks from both servers and clients. By evaluating the system's performance under different attack scenarios, we can ensure that the system can withstand adversarial behaviour and maintain its functionality and integrity.

3. Experimental Evaluation proving the System is Dependable

Throughout our meticulous testing regimen, we delved into specific scenarios to thoroughly evaluate the project's resilience against Byzantine faults.

In the context of integrating the Istanbul Byzantine Fault Tolerance (IBFT) algorithm into a blockchain system, as detailed in the referenced PDF document, a comprehensive suite of eight tests was devised to validate the system's resilience and accuracy, especially in handling Byzantine faults. These tests are crucial for ensuring that the blockchain maintains its integrity and reliability, even in the presence of malicious actors or faulty nodes within the network.

The first two tests focused on the system's ability to correctly verify balance amounts despite interference from Byzantine actors, be they the leader (Test 1) or a non-leader server (Test 2). The success of these tests hinges on the system's reliance on a quorum of messages, ensuring that a single deceptive input cannot skew the consensus.

Tests 3 and 4 examined the system's capability to accurately process transfer amounts, even amidst disruptive actions by Byzantine leaders and servers, respectively. The key to their success lies in the system's method of storing and validating client transfer messages against an array. This validation process allows the system to disregard unrecognised or malicious messages, maintaining accuracy in transfer amounts.

Similarly, Tests 5 and 6 addressed the verification of the correct transfer recipient under the influence of Byzantine leaders and servers, following the same logic and validation mechanism as Tests 3 and 4. This approach guarantees that transactions reach their intended recipients without being misled by faulty or malicious nodes.

Test 7 presents a unique scenario where a server leader attempts to disrupt the round change process by sending messages with incorrect round information. This test is pivotal for ensuring the system's robustness in maintaining protocol integrity during leadership transitions and round changes.

Finally, Test 8 targets the verification of the correct transfer source after interference from a Byzantine client. This test is crucial for affirming the authenticity of transaction origins, relying on the verification of public keys to match transfer sources with their corresponding senders.

Together, these eight tests form a comprehensive evaluation framework for the blockchain system's resilience against Byzantine faults, ensuring its reliability, integrity, and security in a wide range of adversarial conditions. Through these rigorous validations, the system demonstrates its capability to uphold transaction accuracy and consensus integrity, even in the face of sophisticated disruptions.

4. Conclusion

In conclusion, the HDS Serenity (HDS2) project is dedicated to advancing the capabilities of permissioned blockchain systems by focusing on key objectives. These include the implementation of a secure cryptocurrency application, enhancing Byzantine fault tolerance, and conducting experimental validations of correctness, performance, and resilience against attacks. By addressing critical challenges such as security, dependability, and resilience, the project aims to refine the application

semantics and guarantees of permissioned blockchain technology. Through meticulous design, implementation, and testing, HDS2 strives to contribute to the evolution of permissioned blockchain systems, pushing the boundaries of what is possible in decentralised, secure, and reliable transactional networks.